

Solution parallèle pour un problème de dynamique des populations

Guillaume Latu

*LaBRI, UMR CNRS 5800, Université Bordeaux I & ENSERB
351, cours de la Libération, 33405 Talence
latu@labri.u-bordeaux.fr*

*RÉSUMÉ. Un macro-parasite du bar, *Diplectanum aequans*, constitue un élément pathogène en aquaculture. Un modèle mathématique discret [LAN 95, SIL 97, BOU 98] a été élaboré permettant d'en rendre compte. La simulation numérique qui en est issue permet de reproduire certaines des dynamiques de population observées sur le terrain, et fournit de nouveaux éléments pour les comprendre. Les temps d'exécution du simulateur étant rédhibitoires, une mise en œuvre parallèle du simulateur est incontournable. Dans ce papier, nous présentons le problème biologique et la simulation numérique associée, puis notre solution parallèle avec des résultats expérimentaux sur IBM SP2. Les temps d'exécution ont été réduits et la précision des calculs améliorée, permettant ainsi de reproduire des dynamiques observées sur le terrain qui n'étaient pas accessibles avec le simulateur précédent.*

*ABSTRACT. *Diplectanum aequans*, a macroparasite of the sea bass, can cause pathological problems especially in fish farms. A discrete mathematical model [LAN 95, SIL 97, BOU 98] describes the demographic strategy of such fish and parasite populations. Numerical simulations based on this model mimic some of the observed dynamics, and supply hints about the global dynamics of this host-parasite system. Parallelization is required because execution times of the simulator are too long. In this paper, we introduce the biological problem and the associated numerical simulator. Then, a parallel solution is presented, with experimental results on an IBM SP2. The increase in speed has allowed us to improve the accuracy of computation, and to observe some dynamics that the previous simulator had not produce.*

MOTS-CLÉS : système hôte-parasite, simulation numérique, BLAS, algorithmique parallèle.

KEY WORDS : host-parasite system, numerical simulation, BLAS, parallel algorithmic.

1. Introduction

1.1. *Présentation du modèle biomathématique*

Les systèmes hôte-parasite comportent des mécanismes si complexes que les modèles déterministes généraux se révèlent insuffisants pour décrire chaque système dans sa spécificité ; c'est le cas des modèles basés sur des séries infinies d'équations différentielles [SIL 97]. L'un des problèmes centraux à traiter est d'intégrer les processus de recrutement des parasites par les hôtes influençant leur distribution sur ces hôtes. Des travaux dans ce domaine sont d'autant plus nécessaires que des problèmes de pathologie parasitaire et d'épidémiologie apparaissent constamment avec le développement récent des différentes formes d'élevages en environnement marin. Dans des articles récents, Langlais et Silan [LAN 95, BOU 98] ont décrit un modèle à la fois stochastique et déterministe. Des phénomènes fins tels que des hétérogénéités de la population parasitaire y sont reproduits. A partir de ce modèle discret, des études quantitatives via la simulation numérique sont possibles, ce qui permet la validation du modèle. De nombreux paramètres sont pris en compte afin d'obtenir une simulation très fine.

1.2. *De la dynamique des populations à la simulation numérique*

Le modèle biomathématique utilisé dans ce travail est issu d'une modélisation déterministe de dynamique de population. Ce type de modélisation est habituellement basé, soit sur un système d'équations différentielles non linéaires, soit sur une discrétisation d'un tel système [MUR 89]. Les problèmes traités dans ce cadre reposent généralement sur un nombre restreint de paramètres ; ils ont pour objectif principal de reproduire des dynamiques de populations en exploitant peu de données quantitatives sur les populations étudiées. Ces modèles font abstraction de certains processus se déroulant à l'échelle des individus, et cherche à capturer les phénomènes macroscopiques au niveau des populations. Une approche radicalement différente, qui connaît actuellement un développement soutenu, est basé sur des modèles individus centrés aussi appelés multi-agents. Le principe est de reproduire les interactions entre des individus (ou agents) et l'environnement, ou bien entre les individus entre eux selon certaines règles spécifiques aux populations étudiées [HRA 97, BAV 92]. Ces modèles se proposent de reproduire fidèlement des processus élémentaires naturels, mais de ce fait conduisent parfois à des coûts de calcul élevés [DEE 96].

La qualité de réalisme des simulations diffère donc selon le type d'approche envisagée [BED 99]. Le modèle utilisé dans nos travaux est discrétisé en temps mais utilise de nombreuses observations de terrain concernant les hôtes et parasites considérés. Cette approche est donc hybride par rapport à celles qui viennent d'être présentées. Elle permet un niveau de détail fin, notamment grâce aux concepts de structuration en âge des parasites [MUR 89, p. 29], et à celui de l'agrégation prioritaire des parasites

sur les hôtes les plus parasités. Le coût de simulation de ce modèle déterministe et finement paramétré est élevé.

Des modèles individus centrés ont été implémentés sur machine parallèle [DEE 96], car certains systèmes complexes nécessitent de grandes puissances de calcul disponibles uniquement sur ces machines [LOR 95]. Le parallélisme a aussi été utilisé en dynamique des populations pour des problèmes intégrant une structuration spatiale, en utilisant des automates cellulaires [MAN 93]. On trouve dans la littérature peu de références à des simulateurs réalistes qui utilisent des modèles déterministes en temps discret [CHA 99], et encore moins qui sont implantés sur machine parallèle [ALB 99].

1.3. Objectifs

Le premier objectif de ces recherches est de découvrir l'importance relative de chacun des mécanismes biologiques qui interviennent dans le système hôte-parasite. Pour cela, on ajuste les paramètres pour obtenir des simulations réalistes. Le second objectif est d'étudier la sensibilité et la stabilité du modèle en tenant compte des conditions initiales. Le troisième objectif est enfin de rechercher et tester des méthodes prophylactiques en aquaculture (et plus généralement en épidémiologie).

Un simulateur a déjà été réalisé antérieurement, mais ses temps d'exécution pour certaines simulations peuvent durer plus d'un mois sur station de travail. Il est très important de réduire ces temps, afin d'avoir un outil utilisable. Ce document présente les principaux éléments qui ont permis l'élaboration de la version parallèle du simulateur. La complexité des calculs et l'optimisation séquentielle sont d'abord explicitées. Ensuite, la solution algorithmique et ses performances sont exposées. Enfin, nous présentons un aperçu des résultats biologiques obtenus. Ce papier complète les résultats présentés dans [LAN 99].

Ce projet est issu d'une collaboration interdisciplinaire : dynamique des populations (UPR CNRS de Sète), mathématiques appliquées (MAB de l'Université Bordeaux II), informatique (LaBRI de l'Université Bordeaux I). Ce travail est supporté par le programme « Environnement, Vie et Sociétés » et le GDR « Architecture, Réseaux et système, et Parallélisme » (thème iHPerf) du CNRS.

2. Description du problème informatique

2.1. Simulation numérique, principes

La simulation numérique a pour but principal de décrire l'évolution de deux populations, les hôtes et les parasites, durant une année. Les conditions environnementales (comme la température de l'eau) et les données biologiques concernant les parasites sont gérées dans les simulations. Ce que l'on obtient d'une simulation, ce sont les va-

leurs des différentes variables internes à chaque pas de temps (le pas de temps est de $\Delta t = 2$ jours).

On considère que la mortalité des hôtes ne dépend que de leur charge parasitaire ; leur démographie naturelle est négligée pour des raisons d'échelle de temps et de situation en élevage qui sert de motivation à l'étude. Une structuration en âge fine de l'ensemble des parasites sur les hôtes a été mise en place. Cela sert à distinguer les jeunes parasites des adultes. En effet, les parasites adultes seuls pondent des œufs, tandis que les jeunes et les adultes contribuent ensemble à la charge parasitaire. Dans cette étude, on considère 9 classes d'âge pour les parasites jeunes et 1 classe de parasites adultes, autrement dit les parasites deviennent adultes en 18 jours. Les classes d'âge de jeunes rassemblent tous les parasites qui à un certain temps t sont tous d'âge $k \Delta t$ avec $k \in [1, K - 1]$, $K = 10$. La classe des parasites adultes comporte tous les parasites d'âge supérieur ou égal à $K \Delta t$.

Dans la modélisation des systèmes hôte-parasite, une information est particulièrement importante : la distribution des parasites sur les hôtes. En effet, l'évolution du système et certains mécanismes principaux font intervenir cette donnée. Les modèles déterministes continus contiennent moins de paramètres et certaines hypothèses simplificatrices amènent à ce que la forme de cette distribution soit assez figée tout au long des calculs (voir la synthèse faite dans [BOU 97]). On s'interdit alors de laisser le système libre et de constater après coup la distribution des parasites sur les hôtes. Le modèle discret utilisé ici a cet atout de reproduire les mécanismes sans préjuger de la forme de la distribution. En contrepartie, la quantité de calcul à effectuer devient importante ce qui introduit un problème de mise en œuvre informatique.

2.2. Données importantes du modèle biomathématique

La distribution et le recrutement des parasites sur les hôtes, on l'a souligné, sont très importants ; dans ce modèle on fait intervenir les classes d'âge de parasites au sein de cette distribution. La structure de donnée qui en résulte est plus volumineuse que les autres données du modèle : elle est tridimensionnelle si l'on exclut la dimension temporelle. Soit $N_*(*, *, t)$ cette structure avec $N_k(l, i, t)$ le nombre d'hôtes ayant l parasites, i parasites étant plus vieux que $k \Delta t$ au temps t . Ainsi, on tient compte de la structuration en âge de ces parasites (variable k) ; la population de poissons est structurée en sous ensembles qui ont le même nombre de parasites (variable l) ; la distribution des parasites sur un poisson est donnée par la variable i . La mise à jour de $N_*(*, *, t)$ est généralement le calcul le plus coûteux de l'algorithme.

La distribution des parasites sur les hôtes, sans tenir compte de l'âge de ces parasites, doit être connue dans l'algorithme. On pose donc $R(l, t)$ le nombre d'hôtes ayant l parasites au temps t . On a la relation

$$\forall t, \forall k, \forall l, \quad \sum_{i=0}^l N_k(l, i, t) = R(l, t) \quad ,$$

qui est invariante selon k . Elle signifie que pour tout k , le nombre d'hôtes ayant l parasites est égal à la somme des nombres d'hôtes ayant l parasites en tout, et i parasites d'âge supérieur ou égal à $k \Delta t$ lorsque l'on envisage tous les i possibles.

2.3. Mécanismes du modèle

Le cycle de vie du parasite *Diplectanum aequans* est schématisé dans la figure 1. Les différentes étapes de ce cycle sont expliquées dans la suite de la section.

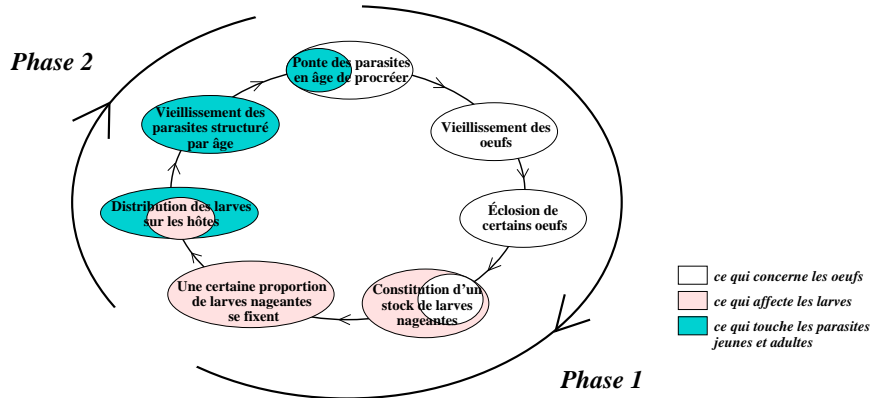


Figure 1. Cycle du développement parasitaire

On peut distinguer deux phases différentes dans la mise à jour des variables du modèle qui interviennent à chaque pas de temps. Dans la première, les variables et fonctions qui concernent les œufs de parasites, les larves nageantes et les larves fixées sont mises à jour individuellement et séquentiellement. Dans la seconde phase, les structures de données $N_*(*, *, t)$ et R qui représentent les distributions des parasites sur les hôtes sont mises à jour. Les formules mathématiques qui sont alors utilisées modélisent plusieurs phénomènes à la fois (mortalité des hôtes, des parasites, recrutement des parasites). Elles utilisent des fonctions de probabilité qui modélisent des mécanismes spécifiques. On a donc en premier lieu une série de petites opérations à effectuer (un mécanisme à chaque fois), et dans un deuxième temps des mises à jour très coûteuses faisant intervenir de multiples mécanismes.

2.3.1. Phase 1 (cf. figure 2)

La première phase est constituée de 5 étapes :

1. La ponte des œufs par les parasites adultes.
2. Le vieillissement des œufs. On définit des classes d'âge pour les œufs ; d'un pas de temps au suivant, les œufs passent d'une classe d'âge à l'autre afin de simuler

le vieillissement. Lors du passage d'une classe à l'autre une certaine proportion de larves meurt (le taux de mortalité dépend de la température).

3. Selon la température, les œufs ont un certain temps d'incubation, qui est suivi de l'éclosion. Un contingent de larves nageantes est alors présent dans le milieu aquatique durant un pas de temps ($\Delta t = 2$ jours).

4. Ce contingent de larves se voit renforcé d'un plus ou moins grand nombre de larves nageantes extérieures contenues dans l'eau alimentant le bassin (selon le mois de l'année). En fait, cet apport extérieur de larves nageantes est le facteur déclenchant du système hôte-parasite.

5. Une partie des larves nageantes se fixent sur les poissons. Ceci est déterminé par le facteur de transmission qui est une fonction $T(H(t))$ ayant comme paramètre le nombre d'hôtes $H(t)$ présents dans le bassin. Le nombre de larves fixées est égal au nombre de larves nageantes que multiplie le facteur de transmission.

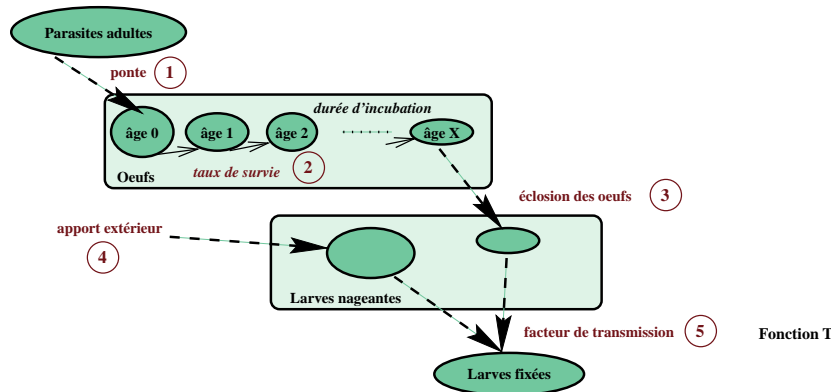


Figure 2. Schéma de la phase 1

2.3.2. Phase 2

La deuxième phase se redécompose en deux parties : tout d'abord l'élaboration de fonctions auxiliaires de probabilité qui dépendent du contexte environnemental, puis le calcul avec ces fonctions de la mise à jour de $R(*, t)$ et $N_*(*, *, t)$.

Construction des fonctions auxiliaires (cf. figure 3)

Cette première partie est organisée en 3 étapes :

1. On aborde un mécanisme assez complexe, le recrutement. Les larves fixées sont réparties sur les hôtes en trois temps :

(a) Un des paramètres de la simulation est une valeur *palier*. Un hôte qui a au dessous de *palier* parasites est considéré comme bien portant, à l'inverse de celui qui en a plus. Une fonction F permet de spécifier que la fraction d'hôtes ayant plus de

palier parasites va accueillir relativement plus de parasites que ceux qui sont bien portants (phénomène d'agrégation des parasites sur les hôtes infestés et affaiblis).

(b) Une deuxième fonction intervient dans le recrutement: $f(l, t)$ qui indique le taux de recrutement moyen d'un hôte ayant l parasites. Soit $C(l, t)$ la classe des hôtes ayant un même nombre de parasites l . La fonction f va indiquer que chaque classe d'hôtes ayant moins de *palier* parasites recrute un nombre identique de larves fixées: $\forall l \leq \text{palier}, f(l, t) = \text{constante}$ (les différentes classes sont parasitées de la même manière). Elle permet aussi d'exprimer qu'au-delà de *palier* parasites les classes d'hôtes recrutent d'autant plus de parasites que le nombre de parasites l sur les hôtes est grand ($f(l, t)$ est une fonction croissante de l pour l supérieure à *palier*). Il s'agit là encore d'un phénomène d'agrégation sur les hôtes affaiblis.

(c) A partir de ces deux fonctions, on forme une troisième fonction φ qui intervient directement dans les formules de mises à jour de $R(l, t)$ et $N_k(l, i, t)$. Plus précisément, $\varphi(j, l, t)$ est la probabilité que j larves soient recrutées par un hôte ayant l parasites au temps t .

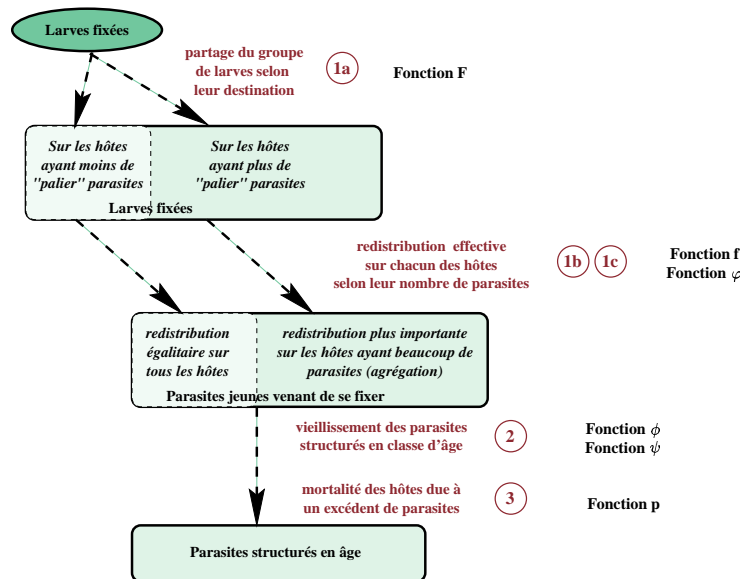


Figure 3. Schéma de la première partie de la phase 2

2. Le vieillissement des parasites distribués sur les hôtes est décrit à l'aide de deux fonctions ϕ et ψ . Pour schématiser, la cohorte¹ de parasites se situant dans la classe de parasites d'âge $k \Delta t$ passe avec un certain taux de survie dans la classe d'âge $(k+1) \Delta t$ au pas de temps suivant, et ceci à l'exception de la dernière classe d'âge $k = K = 10$

1. En démographie, cohorte désigne l'ensemble des individus nés au cours d'un même intervalle de temps.

où les parasites s'accumulent. Cette classe est celle des parasites adultes (toutes les autres classes qui la précède sont donc celles des jeunes). La fonction $\phi(u, l)$ désigne la probabilité que u parasites meurent sur un hôte ayant l parasites. D'autre part, étant donné un hôte qui a l parasites, i étant plus vieux que $k\Delta t$, $\psi(v, i, u, l)$ est la probabilité que v parasites plus vieux que $k\Delta t$ meurent, sachant que u parasites meurent sur cet hôte.

3. La mort des hôtes trop parasités est décrite par la fonction $p(l)$, qui représente la probabilité pour un hôte de survivre à l parasites pendant Δt . A partir d'un certain nombre de parasites, plus l'hôte a de parasites, plus il est probable qu'il meure (la probabilité de survie décroît rapidement vers 0).

Avant de préciser la mise à jour de $R(*, t)$ et de $N_*(*, *, t)$, qui constitue la deuxième partie de la phase 2, on récapitule les notations principales utiles dans la suite de ce papier et on en introduit trois dernières :

- $R(l, t)$: nombre d'hôtes ayant l parasites au temps t ;
- $N_k(i, l, t)$: nombre d'hôtes ayant l parasites, i étant plus vieux que $k\Delta t$;
- $p(l)$: probabilité pour un hôte de survivre à l parasites pendant Δt ;
- $\varphi(j, l, t)$: probabilité que j larves soient "recrutées" par un hôte ayant l parasites au temps t ;
- $\phi(u, l)$: probabilité que u parasites meurent sur un hôte ayant l parasites ;
- $\psi(v, i, u, l)$: étant donné un hôte qui a l parasites, i étant plus vieux que $k\Delta t$, probabilité que v parasites plus vieux que $k\Delta t$ meurent, sachant que u parasites meurent sur cet hôte ;
- K : nombre de classes d'âge de parasites utilisées dans le modèle ;
- $S_{max}(t)$: nombre maximal de parasites observés sur un hôte au temps t ;
- l_{letal} : nombre maximal de parasites que l'hôte peut supporter sans mourir (au-delà de cette limite, un hôte ne peut survivre plus d'un pas de temps) ;
- $S(t) = \min(S_{max}(t), l_{letal})$: lorsque $S_{max}(t)$ est supérieur à l_{letal} , on considère que tous les hôtes ayant au-delà de $S(t)$ parasites seront morts le pas de temps suivant.

Deux valeurs sont centrales dans la suite de l'article car elles sont paramètres de la complexité des calculs les plus coûteux : la valeur $S(t)$ qui vient d'être présentée et qui est majorée par le nombre minimum de parasites qui est léthal pour un hôte (actuellement on considère que $S(t) \leq l_{letal}$ avec $l_{letal} = 700$ parasites), ensuite la valeur K introduite dans la section 2.1 qui est le nombre de classes d'âge utilisées ($K = 10$).

Mise à jour de $R(l, t)$

Considérons maintenant un hôte avec l parasites au temps $t+\Delta t$. Durant l'intervalle de temps de t à $t+\Delta t$, il peut avoir recruté j larves alors que u parasites mourraient de mort naturelle (voir figure 4). Donc au pas de temps précédant t , cet hôte avait $l+u-j$

parasites avec $0 \leq j \leq l$ parce qu'aucune des j larves n'est morte durant la phase de recrutement ; de plus, il a survécu $l+u-j$ parasites et l'on a $0 \leq l+u-j \leq S(t)$. Finalement :

$$R(l, t + \Delta t) = \sum_{j=0}^l \sum_{u=0}^{S(t)-l+j} R(l+u-j, t) \cdot \varphi(j, l+u-j, t) \cdot \phi(u, l+u-j) \cdot p(l+u-j). \quad (1)$$

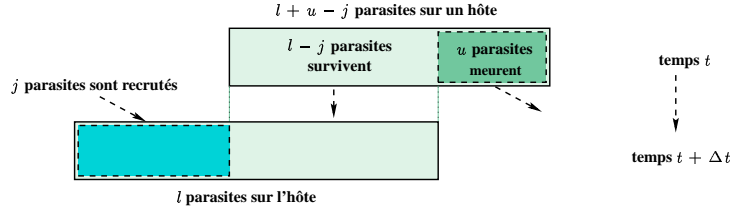


Figure 4. Recrutement et mortalité des parasites sur un hôte

Mise à jour de $N_1(i, l, t)$

Considérons un hôte ayant l parasites parmi lesquels i sont plus vieux que Δt . Au pas de temps t cet hôte avait au moins i parasites à cause du processus de vieillissement, c'est à dire $i + u$ parasites ; entre t et $t + \Delta t$, u parasites sont morts d'une mort naturelle alors que $l - i$ étaient recrutés. De plus, l'hôte doit survivre à une charge de $i + u$ parasites (voir figure 5). On en déduit que :

$$N_1(i, l, t + \Delta t) = \sum_{u=0}^{S(t)-i} R(i+u, t) \cdot \varphi(l-i, i+u, t) \cdot \phi(u, i+u) \cdot p(i+u).$$

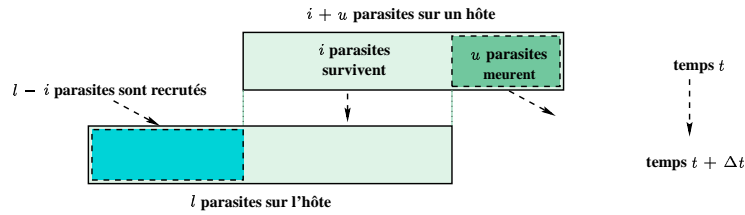


Figure 5. Recrutement et mortalité des parasites sur un hôte

Mise à jour de $N_k(l, i, t)$

Considérons maintenant, en $t + \Delta t$ un hôte ayant l parasites parmi lesquels i sont plus vieux que $k\Delta t$, $2 \leq k \leq K$ (voir figure 6). Au temps t cet hôte avait au moins i

parasites plus vieux que $(k - 1) \Delta t$, c'est à dire $i + \nu$ parmi lesquels ν sont morts de mort naturelle entre t et $t + \Delta t$; la charge totale de parasites au temps t était $m + u$ parmi lesquels u meurent, $0 \leq \nu \leq u$, ce qui conduit à un recrutement de $l - m$ larves. Enfin l'hôte survit à une charge de $m + u$ parasites. Ceci est illustré dans la figure 6. On a donc (pour $2 \leq k \leq K$, $0 \leq i \leq l$, $0 \leq l \leq S(t)$)

$$N_k(l, i, t + \Delta t) = \sum_{m=i}^l \sum_{u=0}^{S(t)-m-u} \sum_{v=0}^u N_{k-1}(m+u, i+v, t) \cdot \varphi(l-m, m+u, t) \cdot \phi(u, m+u) \cdot \psi(i, i+v, m, m+u) \cdot p(m+u) . \quad (2)$$

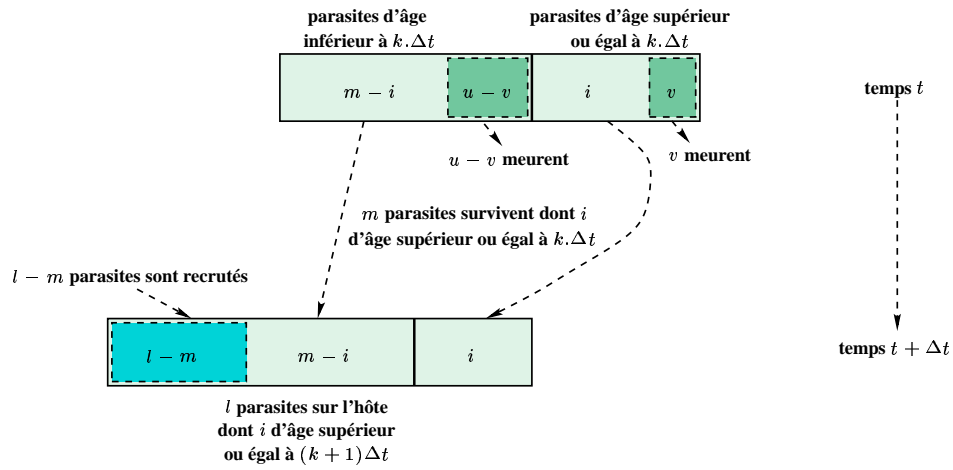


Figure 6. Recrutement et mortalité des parasites structurés en âge sur un hôte

Le temps cumulé de l'ensemble des mises à jour de $N_*(*, *, t)$ et $R(*, t)$ peut représenter près de 99 % du temps total de simulation. Le travail visant à raccourcir les temps d'exécution de l'application a donc surtout porté sur cette partie de la simulation.

3. Analyse séquentielle

Un objectif important est d'évaluer puis de diminuer le coût de la mise à jour de $N_*(*, *, t)$. Dans la suite de ce document, nous allons préciser comment effectuer cette mise à jour de $N_*(*, *, t)$ en $N_*(*, *, t + \Delta t)$. Précisons que lorsque l'on réalise une mise à jour t est constant.

3.1. Evaluation de la complexité

Premièrement, il est intéressant de supprimer les différents appels de fonctions de l'équation (2). En effectuant un certain nombre de précalculs, il est possible de se ramener à une formule qui n'utilise que des accès tableau. Notamment, en effectuant une reformulation mathématique de la fonction ψ on peut réduire son coût de calcul à une seule multiplication. Pour cela, on précalcule en début de simulation deux tableaux ψ_1 et ψ_2 qui ont chacun deux dimensions. En utilisant un accès à ces tableaux calculés à partir des 4 arguments de la fonction ψ , on a alors $\psi(i, i + v, m, m + u) = (\psi_1 \cdot \psi_2)(i, i + v, m, m + u)$. D'autres précalculs sont effectués pour les différentes fonctions qui interviennent dans les formules de mise à jour. D'autre part, la place prise en mémoire par les différents tableaux issus du précalcul prennent en mémoire 10 Mo lorsque les simulations se font en réel simple précision. Deuxièmement, l'intervention des sommes et des factorisations simples réduit le nombre total d'opérations à réaliser. Après ces transformations et les changements d'indice $c = m + u$ et $d = i + v$, l'équation (2) devient :

$$\forall l \in [0, S(t)], \forall i \in [0, l], \forall k \in [2, K],$$

$$N_k(l, i, t + \Delta t) = \sum_{c=i}^{S(t)} p(c) \cdot \sum_{m=i}^{\min(c, l)} \varphi(l - m, c, t) \cdot g(l, i, m, c) , \quad (3)$$

où

$$g(l, i, m, c) = \underbrace{\phi(c - m, c) \cdot \sum_{d=i}^{c - m + i} N_{k-1}(c, d, t)}_2 \cdot \underbrace{(\psi_1 \cdot \psi_2)(d - i, d, c - m, c)}_1 . \quad (4)$$

Chaque fonction correspond maintenant à un accès tableau, et on peut compter le nombre d'opérations $W(S(t), K)$ nécessaires pour évaluer (3). La structure $N_*(*, *, t)$ qu'il faut mettre à jour est de dimension $\Theta(K S(t)^2)$; le calcul de chaque élément de cette structure est en $\Theta(S(t)^3)$. On trouve pour la complexité polynomiale un terme de plus haut degré de $\frac{3K}{40} S(t)^5$. On a alors $W(700, 10) = 115 \text{ TFLOP}$, ce qui n'est pas un coût raisonnable pour une seule mise à jour de $N_*(*, *, t)$ (une simulation requiert 183 mises à jour).

3.2. Optimisation et utilisation des BLAS

On peut aller plus avant dans la factorisation en évitant les redondances de calcul. Des parties de l'équation (3) sont en fait réévaluées plusieurs fois dans certaines combinaisons des variables l, i, k . Dans l'équation (4), on voit que le terme **1** ne dépend pas de la variable k ce qui implique que pour différents k il est calculé plusieurs fois. A l'aide d'un remaniement de l'algorithme, on élimine le recalcul. Mais

combien gagne-t-on ainsi ? Comme la multiplication $\psi_1.\psi_2$ est faite une fois seulement pour l'ensemble des k , il reste dans la boucle la plus interne de l'algorithme deux opérations en moyenne (une multiplication plus une addition), à la place de trois opérations (deux multiplications plus une addition). Ainsi, la nouvelle complexité de l'algorithme représente à peu près $\frac{2}{3}$ de la complexité précédente. De manière similaire, il est intéressant de noter que le terme **2** de (4) est réévalué lorsque l varie sur son intervalle de définition. L'algorithme est à nouveau remanié pour éviter cela. La complexité est divisée par $\frac{S(t)}{5}$ donnant une complexité polynomiale diminuée d'un ordre de grandeur. On a donc finalement une complexité dont le terme de plus haut degré est $\frac{K}{4}S(t)^4$ et $W(700, 10) = 560_{\text{GFLOP}}$. Le coût d'une mise à jour de $N_*(*, *, t)$ a donc été réduit de manière très notable, et on peut maintenant envisager la vectorisation de l'algorithme.

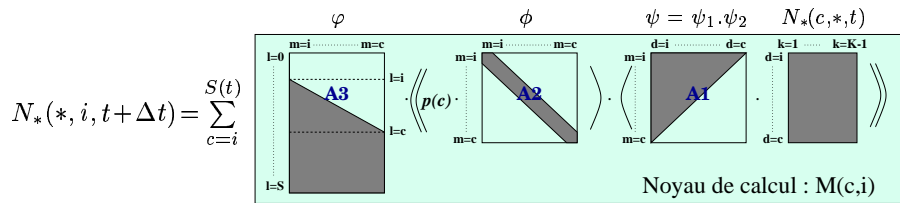


Figure 7. Mise à jour de $N_*(*, *, t)$ utilisant une expression matricielle

Les BLAS [AND 95] conduisent à une réduction des temps d'exécution pour les calculs vectoriels. En modifiant l'ordre des boucles de calcul, on obtient naturellement plusieurs formulations vectorielles de l'algorithme, mais une seule n'utilise que des opérations matrice-matrice. C'est sous cette forme que l'algorithme utilisant les BLAS 3 conduit aux meilleures performances. Cependant, il y a une nouvelle étape dans l'algorithme qui consiste à construire des matrices intermédiaires A_1, A_2, A_3 à l'aide respectivement des tableaux (ψ_1 et ψ_2), ϕ , φ . Ceci nécessite un certain temps et est optimisé le plus possible. La mise à jour de $N_*(*, *, t)$ peut maintenant être réécrite² comme l'illustre la figure 7. Les matrices intermédiaires contiennent par construction beaucoup de termes nuls qui sont représentés par la couleur gris clair dans la figure 7. En effet, les dépendances entre les différents indices des boucles de calcul impliquent que certains espaces ne sont pas couverts par les indices, ce qui introduit des termes nuls de construction dans les matrices intermédiaires. En reprenant par exemple l'équation (4), on note que l'indice d varie entre i et $c-m+i$; ceci explique que pour $d > c - m + i$ dans la matrice A_1 les coefficients sont à zéro. Par définition, $M(c, i)$ représentera le noyau de calcul qui contient les multiplications de matrices se situant à l'intérieur de la somme. Il est possible de comparer des ordres différents dans les multiplications de matrices mais une seule solution est asymptotiquement la plus économique en nombre d'opérations : $A_3 \cdot (p(c) \cdot A_2) \cdot (A_1 \cdot N_*(c, *, t))$. On peut maintenant étudier la version parallèle de l'algorithme.

2. La notation * signifie que l'on considère tous les éléments d'une certaine dimension.

4. Solution parallèle

4.1. Introduction

Pour conserver la performance des calculs BLAS, il faut envisager de paralléliser en préservant le noyau de calcul $M(c, i)$. L'algorithme auquel on est conduit est celui de la figure 8.

Si l'on considère l'ensemble des tâches $M(c, i)$ qui sont à réaliser, l'algorithme doit être considéré comme étant à grain moyen avec $(S(t) + 1)(S(t) + 2)/2$ noyaux à distribuer sur les P processeurs. Par contre, si l'on considère la parallélisation d'une seule des boucles de l'algorithme, il n'y a plus que $S(t) + 1$ tâches à répartir, d'où un parallélisme à gros grain. Ces différentes possibilités ont été envisagées, mais nous présentons dans ce papier uniquement la parallélisation de la boucle en i .

```

Procédure Mise_à_jour_de_N(S(t)) {
  Pour i:=0 à S(t) faire {
    . Pour c:=i à S(t) faire {
    . . Initialisation des matrices  $A_1, A_2, A_3$ ;
    . .  $Aux \leftarrow A_3 \cdot (p(c) \cdot A_2) \cdot (A_1 \cdot N_*(c, *, t))$ ; /* tâche  $M(c, i)$  */
    . .  $N_*(*, i, t + \Delta t) \leftarrow N_*(*, i, t + \Delta t) + Aux$ ;
    . . }
    . }
  }

```

Figure 8. Algorithme de mise à jour de $N_*(*, *, t)$

4.2. Distribution des calculs et des données

Par la suite, la donnée $N_*(c, *, t)$ est appelée «*plaque ligne* (c, t)», et $N_*(*, i, t)$ est appelée «*plaque colonne* (i, t)». A l'itération ($i = i_0, c = c_0$) de l'algorithme (figure 8), la plaque ligne (c_0, t) est utilisée pour calculer une contribution pour la plaque colonne ($i_0, t + \Delta t$). La plaque colonne ($i_0, t + \Delta t$) dépend de l'ensemble des plaques lignes (c, t) pour lesquelles $S(t) \geq c \geq i_0$ (voir figure 11). Les données sont donc les plaques lignes et les plaques colonnes constituent les produits. Ceci implique que l'on ne peut pas écraser les données du temps t par celles du temps $t + \Delta t$; on doit disposer de deux structures de données pour $N_*(*, *, t)$ lors de sa mise à jour (une pour le temps t , et l'autre pour le temps $t + \Delta t$). Ceci ne représente que 38 Mo (en réel simple-précision) à distribuer sur la machine parallèle.

En ce qui concerne les calculs, les tâches individuelles M sont indépendantes entre elles. Les tâches $M(*, i_0)$ servent au calcul des contributions à la plaque colonne $N_*(*, i_0, t + \Delta t)$. Il est donc intéressant de placer sur le même processeur les tâches de calcul $M(*, i_0)$ et la matrice qu'elles manipulent $N_*(*, i_0, t + \Delta t)$. Il est possible ainsi de réaliser les additions de matrices $N_*(*, i_0, t + \Delta t)$ de l'algorithme localement. Les indices i_0 sont distribués sur les processeurs avec une distribution en serpent (snake en anglais).

Numéros des processeurs	0	1	2	3
Numéros des plaques colonnes i_0 locales	0	1	2	3
	4	5	6	7
	8	9	10	11
	12	13	14	15
	16	17	18	19
	20	21		

Numéros des processeurs	0	1	2	3
Numéros des plaques colonnes i_0 locales	0	1	2	3
	7	6	5	4
	8	9	10	11
	15	14	13	12
	16	17	18	19
		21	20	

(a) cyclique
(b) serpent

Figure 9. Distributions possibles des $M(*, i_0)$ et plaques lignes $(i_0, t + \Delta t)$ sur les processeurs

Ainsi on obtient un équilibrage de la charge de bonne qualité. Une distribution cyclique peut aussi être envisagée, mais elle donne de moins bons résultats. On place les calculs les plus coûteux en premier sur les processeurs ce qui correspond aux tâches $M(*, i_0)$ avec des indices i_0 faibles (un exemple où $S(t) = 21$ est présenté figure 9).

Il a été décidé de stocker dans la structure $N_*(*, *, t)$ au temps t des plaques lignes, tandis qu'au temps $t + \Delta t$ la structure $N_*(*, *, t + \Delta t)$ calculée contient des plaques colonnes. L'accès aux données lors des calculs est ainsi grandement facilité. Finalement des plaques lignes appartenant à la structure $N_*(*, *, t)$ et des plaques colonnes de la structure $N_*(*, *, t + \Delta t)$ doivent être réparties conjointement sur les processeurs. Tout comme pour les plaques colonnes on utilise une distribution en serpent pour les plaques lignes.

```

Procédure Mise_à_jour_de_N(S(t)) {
  Pour c:=0 à S(t) faire {
    . /* étape de communication */
    . Si (mapC(c)=my_id) {
    . Diffusion totale de  $N_*(c, *, t)$ 
    . } Sinon {
    . Reçoit  $N_*(c, *, t)$ 
    . }
    . /* étape de calcul */
    . Pour i:=0 à c faire {
    . . Si (mapI(i)=my_id) {
    . . . /* tâche  $M(c, i)$  */
    . . . Initialisation des matrices  $A_1, A_2, A_3$ ;
    . . . Aux ←  $A_3 \cdot ((p(c) \cdot A_2) \cdot (A_1 \cdot N_*(c, *, t)))$ ;
    . . .  $N_*(*, i, t + \Delta t) \leftarrow N_*(*, i, t + \Delta t) + \text{Aux}$ ;
    . . . }
    . . }
  }
  Transposition de  $N_*(*, *, t)$  des plaques
  colonnes vers les plaques lignes;
}

```

Figure 10. Mise à jour de $N_*(*, *, t)$

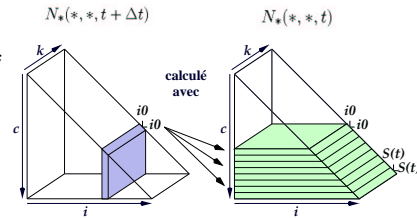


Figure 11. Dépendances des données

L'algorithme a été reformulé pour tenir compte explicitement de ces deux distributions (voir figure 10). Dans celui-ci, la fonction $\text{mapC}(c)$ renvoie le numéro du processeur à qui appartient la plaque ligne (c, t) . De manière similaire $\text{mapI}(i)$ désigne le numéro du processeur propriétaire de la plaque colonne $(i, t + \Delta t)$.

Comme l'algorithme utilise des plaques lignes et génère des plaques colonnes, une étape de transposition est nécessaire pour réaliser plusieurs mises à jour successives. Soit le vecteur $N_*(c_0, i_0, t + \Delta t)$ qui appartient au processeur $\text{mapI}(i_0)$; il doit tout simplement être transmis au processeur $\text{mapC}(c_0)$. La transposition consiste donc à déplacer tous les vecteurs $N_*(c_0, i_0, t + \Delta t)$ vers leur nouvelle place ; ainsi le coût en communication est le même que l'espace mémoire occupé par $N_*(*, *, t)$ c'est-à-dire $\frac{K S(t)^2}{2}$ (soit 19 Mo au maximum pour $S(t) \leq 700$). Cette transposition globale est effectuée à l'aide de la routine `MPI_AllToAll` de la bibliothèque MPI [MPI 94].

Comme on peut le constater dans la boucle en c de l'algorithme, deux étapes peuvent être distinguées : une de calcul, l'autre de communication. La diffusion totale (*broadcast*) utilisée dans l'étape de communication est en pratique anticipée et se déroule en communication asynchrone non bloquante. Ceci permet de tirer profit d'un recouvrement calcul-communication. Autre point, on peut remarquer qu'en faisant tous les calculs concernant la plaque colonne c en une fois (avant de passer à $c + 1$), on augmente la localité temporelle de la donnée $N_*(c, *, t)$.

Les diffusions asynchrones anticipées des plaques lignes sur les P processeurs impliquent un coût de communication de $\frac{K P S(t)^2}{2}$. Cette quantité est très inférieure au coût calculatoire $W(S(t), K)$, et un recouvrement calcul-communication total est réalisable. De même, le coût en communication pour une transposition de $N_*(*, *, t)$ est de $\frac{K S(t)^2}{2}$, et bien que non recouverte, cette étape peut être négligée. L'implémentation confirme ces deux éléments sur nos machines de validation de type IBM SP2 (description des machines en section 5.1). La communication n'altère donc pas les performances du programme parallèle ; nous étudions maintenant l'équilibrage des charges et la scalabilité du programme.

4.3. Coûts en calcul sur chaque processeur

On détermine la complexité $CM(c, i)$ des noyaux de calculs $M(c, i)$ en comptant le nombre de multiplications et d'additions réalisées. Pour cela, on tient compte de l'ordre de multiplication des matrices présentées sur la figure 7.

Posons $\zeta = c - i + 1$, il vient :

- le remplissage de la matrice $A3$ nécessite $\zeta \cdot (\zeta + 1) / 2$ multiplications,
- le calcul de la matrice $B2 = p(c) \cdot A2$ nécessite ζ multiplications,
- le calcul de $B1 = A1 \cdot N_*(c, *, t)$ nécessite $\frac{(K-1) \cdot \zeta \cdot (\zeta+1)}{2}$ multiplications et additions,
- l'opération $B3 = B2 \cdot B1$ nécessite $(K - 1) \cdot \zeta$ multiplications,
- l'opération $B4 = A3 \cdot B3$ nécessite $(K - 1) \cdot (\frac{\zeta \cdot (\zeta+1)}{2} + \zeta \cdot (S(t) - c))$ multiplications et additions,
- l'addition de $B4$ à $N_*(*, i, t + \Delta t)$ nécessite $(K - 1) \cdot (S(t) - i + 1)$ additions.

On en déduit :

$$CM(c, i) = \zeta \cdot \left[\frac{(\zeta+1)}{2} + 1 + (K-1) \cdot (\zeta+1) + (K-1) + (K-1) \cdot (\zeta+1) + 2 \cdot (K-1) \cdot (S(t)-c) \right] + (K-1)(S(t)-i+1) ,$$

d'où

$$CM(c, i) = (c-i+1) \cdot \left[(2K - \frac{3}{2})(c-i+2) + K + (2K-2)(S(t)-c) \right] + (K-1)(S(t)-i+1) .$$

La complexité séquentielle W évoquée précédemment peut être déduite de CM :

$$W(S(t), K) = \sum_{i=0}^{S(t)} \sum_{c=i}^{S(t)} CM(c, i) = \left(\frac{K}{4} - \frac{5}{24} \right) S(t)^4 + o(S(t)^4) .$$

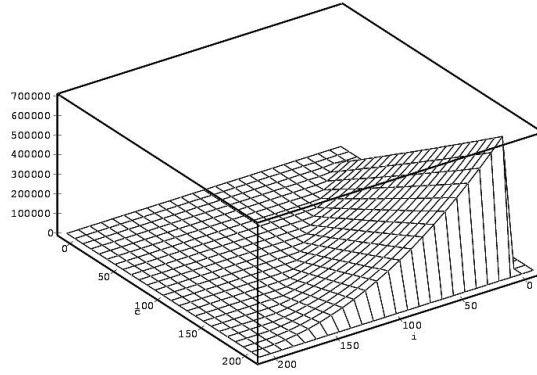


Figure 12. Courbe de $CM(c, i)$ (pour $S(t) = 200$) qui indique les coûts en calcul pour chaque tâche $M(c, i)$

Comme l'illustre la figure 12, la fonction CM est positive, décroissante en i et croissante en c . D'autre part elle est convexe en i et c . Connaissant le nombre d'opérations à réaliser pour chaque tâche M , et la répartition de celles-ci sur les processeurs, nous pouvons calculer combien d'opérations chaque processeur doit réaliser.

Comme le placement des calculs sur les processeurs a été réalisé statiquement (il est décidé une fois pour toute dès le début des calculs de mise à jour de $N_*(*, *, t)$), une étude théorique portant sur l'efficacité est possible. Pour calculer cette efficacité, il faut pouvoir déterminer quel processeur a le plus d'opérations à effectuer, connaître

ce nombre d'opérations, et disposer du coût séquentiel. Par la suite, nous utilisons la proposition suivante démontrée dans [LAT 98] :

Proposition 1 *Soit P le nombre de processeurs ; un nombre fini $(s + 1)$ de tâches élémentaires sont distribuées sur ces processeurs avec une distribution en serpent (on suppose que $s + 1 = 2Pq$ avec $q \in \mathbb{N}^{+*}$). Soit $g(i)$ la fonction donnant le nombre d'opérations à effectuer pour une tâche élémentaire d'indice i . On suppose $g(i)$ convexe décroissante sur $[0, s]$. Soit $charge(p_x)$ la fonction qui donne le coût en nombre d'opérations sur chaque processeur $p_x \in [0, P - 1]$; il vient*

$$charge(p_x) = \sum_{j=1}^{\left(\frac{s+1}{2P}-1\right)} g(2.(j-1).P + p_x) + g(2.j.P - 1 - p_x) .$$

La fonction $charge(p_x)$ est alors positive décroissante sur $[0, P - 1]$.

Dans les problèmes concrets pour lesquels on a $s' + 1$ tâches à distribuer, on ne dispose pas dans tous les cas de l'hypothèse $s' + 1 = 2Pq$. On peut alors appliquer la proposition pour les $s + 1 = \lfloor (s' + 1) / (2P) \rfloor \cdot 2P$ premières tâches. On peut considérer alors que généralement la distribution sur les processeurs des $s' - s$ tâches restantes qui ont les coûts les plus faibles (g est décroissante) ne viennent pas infirmer la conclusion : la fonction $charge(p_x)$ est positive décroissante sur $[0, P - 1]$.

On a distribué sur les processeurs les groupes de tâches $M(*, i)$. Appelons $g(i)$ la complexité d'un tel groupe de tâches ($g(i) = \sum_{c=i}^{S(t)} CM(c, i)$). La fonction $g(i)$ est une somme de fonctions décroissantes convexes en i , elle est donc elle-même une fonction décroissante convexe. Comme on distribue les indices i sur les processeurs selon la méthode du serpent, on se trouve dans les conditions d'application de la proposition 1 avec $s = S(t)$. On en déduit donc directement que la charge des processeurs $[0, P - 1]$ est une suite positive décroissante. Le processeur 0 est donc celui qui a le plus de calculs à réaliser.

Si les temps d'initialisation des matrices intermédiaires, les effets de caches, l'accélération des calculs due au BLAS ne biaisent pas ce résultat, les temps d'exécution des processeurs doivent être de la même manière une fonction décroissante des indices des processeurs. Notre implémentation a cette propriété comme l'illustre la figure 14.

4.4. Efficacité, scalabilité

On va maintenant déduire les conditions pour lesquelles on a une exécution efficace du programme parallèle, en étudiant l'efficacité et la scalabilité de l'algorithme.

Soit la complexité en temps $T_{in}(S(t), K, c, i)$ d'une itération de la boucle la plus interne de l'algorithme de la figure 8 (*c.a.d* de la tâche $M(c, i)$), et $T_{out}(S(t), K, i)$ la complexité de la boucle externe. La complexité totale en temps de la mise à jour de $N_*(*, *, t)$ est notée $T_{seq}(S(t), K)$. Comme évoqué précédemment, $T_{in}(S(t), K, c, i)$

ne correspond pas exactement à la complexité en nombre d'opérations pour trois raisons : l'initialisation des matrices intermédiaires n'est pas négligeable, les effets de caches dus aux réutilisations successives de certaines matrices dans le noyau de calcul ont un impact sur les temps d'exécution, et enfin l'utilisation des BLAS modifie le coût en temps. On en déduit que la complexité en temps $T_{seq}(S(t), K)$ est différente du nombre d'opérations de la mise à jour $W(S(t), K)$, bien que ces deux quantités soient du même ordre asymptotique : on pose $T_{seq}(S(t), K) \approx \beta(K)S(t)^4$. En tenant compte de l'ordre de multiplication des matrices dans $M(c, i)$ (figure 7), on peut essayer d'exprimer le coût en temps $T_{in}(S(t), K, c, i)$. En considérant que K est constant, on observe que le nombre d'opérations $CM(c, i)$ est un polynôme homogène du second degré en i , c , et $S(t)$. Il en est de même pour la complexité en temps bien que les coefficients du polynôme, inconnus et fonction de K , puissent être différents. On a aussi les équations suivantes :

$$T_{out}(S(t), K, i) = \sum_{c=i}^{S(t)} T_{in}(S(t), K, c, i) ,$$

et

$$T_{seq}(S(t), K) = \sum_{i=0}^{S(t)} T_{out}(S(t), K, i) .$$

Soit $T_{//}(S(t), K, P)$ le temps d'exécution parallèle sur P processeurs. Par commodité, on suppose que $S(t) + 1$ est un multiple de $2P$. Le temps d'exécution parallèle correspond au temps d'exécution du processeur 0 comme il a été dit précédemment. La distribution en serpentín conduit à l'expression suivante :

$$T_{//}(S(t), K, P) = \sum_{j=0}^{\frac{S(t)+1}{2P}-1} T_{out}(S(t), K, 2Pj) + T_{out}(S(t), K, 2Pj + 2P - 1) .$$

$T_0(S(t), K, P) = P.T_{//}(S(t), K, P) - T_{seq}(S(t), K)$ constitue dans cette étude (la communication n'engendre pas de surcoût significatif) le surcoût induit par le dés-équilibre des charges. En utilisant la forme polynomiale de $T_{in}(S(t), K, c, i)$, quels que soient ses coefficients inconnus, un calcul formel donne³

$$T_0(S(t), K, P) \approx \Theta(P^2 S(t)^2) .$$

Ceci montre que le surcoût pour chaque mise à jour augmente de manière quadratique en fonction du nombre de processeurs P et de la taille du problème $S(t)$. Soit $\gamma(K)$ un facteur inconnu (on rappelle que $T_{seq}(S(t), K) \approx \beta(K)S(t)^4$) tel que

$$\frac{T_0(S(t), K, P)}{T_{seq}(S(t), K)} \approx \gamma(K) \frac{P^2}{S(t)^2} . \quad (5)$$

3. Le résultat peut être obtenu en utilisant *Maple* par exemple.

L'efficacité (définie dans [GRA 94]) peut être exprimée par

$$E(S(t), K, P) = \frac{1}{1 + \frac{T_0(S(t), K, P)}{T_{seq}(S(t), K)}} , \text{ donc } E(S(t), K, P) \approx \frac{1}{1 + \gamma(K) \frac{P^2}{S(t)^2}} . \quad (6)$$

Pour $K = 10$, des séries d'exécutions tests ont permis de déterminer que $\gamma(10) = 14$. On dispose ainsi d'une fonction qui approxime correctement l'efficacité observée lors des simulations (voir figure 13). Si l'on se fixe une efficacité minimale de 80%, on a :

$$E(S(t), 10, P) > 0.80, \quad 1 > 56 \frac{P^2}{S(t)^2}, \quad \frac{S(t)}{P} > \sqrt{56} \approx 7.5 .$$

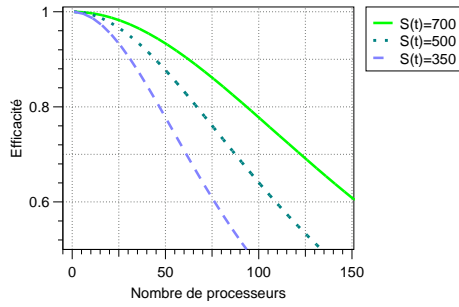


Figure 13. Efficacité approchée avec $\gamma(K = 10) = 14$

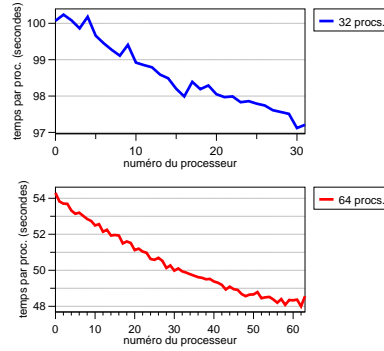


Figure 14. Temps de calcul de la mise à jour de $N_*(*, *, t)$ ($S(t)=700$) pour les simulations du tableau 1 (cf. section 5.1), pour chaque processeur

On a donc obtenu une relation entre $S(t)$ et P qui donne les conditions pour une exécution efficace. Dans le cas de 32 processeurs, la mise à jour parallèle de $N_*(*, *, t)$ est efficace pour $S(t) \in [240, 700]$ (c.a.d des problèmes dont la taille est comprise entre 8 GFLOP et 560 GFLOP); pour 64 processeurs, $S(t)$ doit être dans l'intervalle $[479, 700]$ (problèmes de 123 à 560 GFLOP). Jusqu'à 64 processeurs, on observe que les mises à jour réellement coûteuses sont réalisées avec une efficacité parallèle de bonne qualité.

Comme $T_{seq}(S(t), K) \approx \beta(K)S(t)^4$, et $T_0(S(t), K, P) \approx \Theta(P^2 S(t)^2)$, alors $T_0(S(t), K, P) \ll T_{seq}(S(t), K)$ lorsque P est petit devant $S(t)$. Sous cette hypothèse on en déduit que l'algorithme est à coût optimal :

$$P.T_{//}(S(t), K, P) = \Theta(T_{seq}(S(t), K)) .$$

En réécrivant l'équation (6), on a

$$1 + \gamma(K) \frac{P^2}{S(t)^2} = \frac{1}{E(S(t), K, P)} , \quad (7)$$

soit
$$\frac{P^2}{S(t)^2} = \frac{1 - E(S(t), K, P)}{\gamma(K) E(S(t), K, P)} , \quad (8)$$

d'où
$$S(t) = \sqrt{\gamma(K) \frac{E(S(t), K, P)}{(1 - E(S(t), K, P))}} P . \quad (9)$$

A l'aide de cette formule d'iso-efficacité pour laquelle on fixe $E(K, S(t), P)$ et K , on a plus simplement $S(t) = \Theta(P)$. Lorsque le nombre de processeurs est doublé, $S(t)$ doit de la même manière être doublé pour conserver la même efficacité, ce qui est illustré dans la figure 13 pour $S(t) = 350$ et $S(t) = 700$.

5. Expérimentation numérique

5.1. Performance

Les simulations ont été réalisées sur deux IBM SP2. Au LaBRI⁴, la machine possède 16 Power 2 nœuds fins (66 MHz), avec 64 ou 128 Mo de mémoire. Au CINES⁵, il y a 207 Power 2 Super Chip nœuds fins (120 MHz), avec 256 Mo de mémoire ; un switch TB3 assure l'interconnexion des nœuds. Les résultats présentés dans ce papier proviennent de la machine du CINES. Le code à été développé en FORTRAN 90 avec le compilateur XL Fortran et utilise la bibliothèque MPI (version propriétaire de IBM).

Mise à part la mise à jour de $N_*(*, *, t)$, de nombreuses autres parties du code ont été parallélisées, et il ne reste que quelques opérations séquentielles. Avec 32 à 64 nœuds, un très fort pourcentage du temps total reste consacré au calcul de la mise à jour de $N_*(*, *, t)$ (entre 80 % et 90 %). Pour des simulation réalistes, la valeur de $S(t)$ est 700 durant une grande partie du temps d'exécution de la simulation et il est donc important d'avoir de bonnes performances dans cette configuration (voir tableau

4. Laboratoire Bordelais de Recherche en Informatique.

5. Centre Informatique National de l'Enseignement Supérieur (Montpellier).

1).

	Nombre de processeurs				
	4	8	16	32	64
Temps d'exécution	782.4 s	392.0 s	196.8 s	100.4 s	54.3 s
Efficacité relative	1.00	1.00	0.99	0.97	0.90
Performance (MFLOPS)	179	178	178	174	161

Tableau 1. Temps d'exécution, efficacité relative par rapport à 4 processeurs et nombre moyen d'opérations par seconde et par processeur pour une mise à jour de $N_*(*, *, t)$ ($S(t) = 700$, $K = 10$ i.e. 560 GFLOP)

	Nombre de processeurs			
	8	16	32	64
Temps d'exécution	38633.8 s	19511.4 s	10014.8 s	5534.8 s

Tableau 2. Temps d'exécution pour une simulation complète et coûteuse (qui comprend 88 pas de temps avec $S(t) = 700$)

Une mise à jour de $N_*(*, *, t)$ avec $S(t) = 700$ atteint par exemple une efficacité relative de 90 % sur 64 processeurs. Pour une simulation complète, on remarque que le temps d'exécution est à peu près divisé par deux lorsque le nombre de processeurs est doublé. Cependant, la performance des processeurs n'atteint que le tiers des performances de crête. Il y a deux raisons à cela : premièrement le remplissage des matrices intermédiaires prend du temps, et les multiplications BLAS se font avec une dimension égale à $K-1=9$ (figure 7) ce qui n'autorise pas les plus grandes vitesses de calcul.

5.2. Résultats biologiques

Lorsque l'on change le nombre de processeurs, les calculs parallèles sont ordonnés différemment, ce qui peut causer des erreurs d'approximations. Lorsque l'on utilise la double précision, toutes les variables que l'on observe durant la simulation demeurent identiques sur 7 chiffres significatifs, si le nombre de processeurs change. Pour la simple précision, si le nombre de processeurs est modifié, les résultats ne diffèrent pas sur les 4 premiers chiffres significatifs. La double précision améliore donc notablement la précision des résultats mais le coût calculatoire est 40 % plus grand. La stabilité numérique est importante du point de vue des biologistes et des mathématiciens : les phénomènes observés dans les simulations numériques ne doivent pas provenir d'artéfacts de calculs et le modèle doit être fidèlement reproduit. Néanmoins du point de vue des utilisateurs du simulateur, la simple précision suffit pour l'instant.

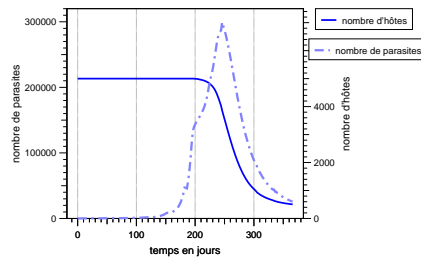


Figure 15. *Etat endémique provisoire à la fin de la simulation*

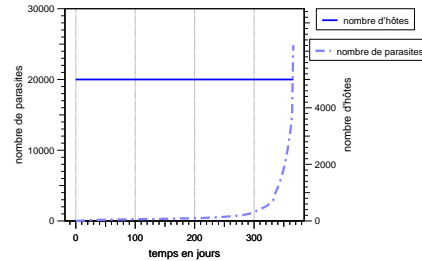


Figure 16. *Croissance de la population parasitaire sans déclenchement de l'épidémie*

Avec le simulateur séquentiel, seulement deux comportements dynamiques étaient observés : premièrement, l'extinction des hôtes après une épidémie ; deuxièmement la disparition des parasites sans aucune mort d'hôte. Avec la version parallèle, les calculs sont plus précis car beaucoup d'approximations permettant d'économiser des calculs ont été supprimées, et les exécutions ne demandant pas beaucoup de temps permettent de simuler beaucoup de dynamiques réalistes. Sur de nouveaux jeux de paramètres, il a été possible d'obtenir deux cas de figure nouveaux qui correspondent à des situations réelles :

- l'épidémie provoquée par les parasites ne provoque pas l'éradication des poissons. Les nombres de poissons et de parasites se stabilisent et on atteint un état endémique provisoire (voir figure 15) ;
- durant toute l'année le nombre de parasites ne cesse de croître, mais le seuil à partir duquel l'épidémie se déclenche n'est jamais atteint (voir figure 16).

Les temps d'exécutions de la simulation qui a permis de réaliser la figure 15 sont présentés dans le tableau 2.

6. Conclusion

Après une analyse des calculs effectués par un simulateur numérique concernant un système hôte-parasite, nous avons pu élaborer une solution parallèle. Une simulation complète et coûteuse ne dure plus que 1 heure 30 sur 64 nœuds contre un mois auparavant. L'analyse de performance a permis de montrer l'efficacité et l'extensibilité (vérifiée jusqu'à 64 nœuds) de l'algorithme parallèle. D'autres distributions des données et des calculs sont actuellement étudiées.

Les simulations réalisées avec le nouveau simulateur ont permis d'observer certaines dynamiques, en accord avec les observations de terrain, qui n'étaient pas accessibles avec l'ancien simulateur. D'autre part, les multiples simulations tentées ont conduit à changer finement les fonctions de modélisation des mécanismes et leurs pa-

ramètres. Ces modifications permettent de réaliser des simulations qui du point de vue biologique sont de plus en plus réalistes.

Dans la configuration actuelle du modèle, changer le nombre de classes d'âge conduirait à des simulations similaires aux simulations actuelles. On étudie un changement dans la structure du modèle permettant de raffiner la simulation lorsque l'on passe de 1 à 40 classes d'âge pour les parasites adultes. D'autre part, les performances obtenues permettent d'envisager un modèle biologique plus stochastique qui demanderait encore plus de calculs.

7. Bibliographie

- [ALB 99] S. AL-BATTRAN, A.J. FIELD, R.L. WILEY, et J. WOODS. « Parallel Simulation of Plankton Ecology ». In *Proceedings of the IASTED International Conference Modelling And Simulation, Philadelphia, USA*, may 1999. p. 259–263.
- [AND 95] E. ANDERSON, Z. BAI, J. BISCHOF, J. DEMMEL, J.J. DONGARRA, J. DUCROZ, A. GREENBAUM, S. HAMMARLING, A. MCKENNEY, S. OSTROUCHOV, et D. SORENSEN. « *LAPACK User's Guide* », SIAM Publications, 2nd édition, 1995.
- [BAV 92] J.M. BAVECO et R. LINGEMAN. « An object-oriented tool for individual-oriented simulation: Host-parasitoid system application ». *Ecological Modelling*, 61:267–286, 1992.
- [BED 99] Mark BEDAU. « Can Unrealistic Computer Models Illuminate Theoretical Biology? ». In A. WU, Ed., *Genetic and Evolutionary Conference Workshop Program, Orlando, Florida*, july 1999.
- [BOU 97] C. BOULOUX. « Modélisation, simulations et analyse mathématique de systèmes hôtes-parasites ». Thèse de doctorat, Université Bordeaux I, décembre 1997.
- [BOU 98] C. BOULOUX, M. LANGLAIS, et P. SILAN. « A marine host-parasite model with direct biological cycle and age structure ». *Ecological Modelling*, 107:73–86, 1998.
- [CHA 99] M.A.J. CHAPLAIN, A.R.A. ANDERSON. « Modelling the growth and form of Capillary networks ». In M.A.J. Chaplain, G.D. Singh, J.C. McLachlan Ed.. « *On growth and form, Spatio-temporal pattern formation in biology* », p. 225–246. Wiley series in Mathematical and computational biology, Simon Levin édition, 1999.
- [DEE 96] E. DEELMAN, T. CARACO, et B. K. SZYMANSKI. « Parallel Discrete Event Simulation of Lyme Disease ». In L. HUNTER et T. KLEIN, Eds., *1996 Pacific Symposium, Hawaii*, p. 191–202. World Scientific Publishing Corp., january 1996.
- [GRA 94] A. GRAMA, A. GUPTA, V. KUMAR, et G. KARYPIS. *Introduction to Parallel Computing: Design and Analysis of Algorithms*. Benjamin/Cummings Publishing Company, Redwood City, 1994.
- [HRA 97] P.T. HRABER, T. JONES, et S. FORREST. « The ecology of Echo ». *Artificial Life*, 3(3):165–190, 1997.
- [LAN 95] M. LANGLAIS et P. SILAN. « Theoretical and mathematical approach of some regulation mechanisms in a marine host-parasite system ». *Journal of Biological Systems*, 3(2):559–568, 1995.
- [LAN 99] M. LANGLAIS, G. LATU, J. ROMAN, et P. SILAN. « Parallel numerical simulation of a marine host-parasite system ». In *Europar'99 Parallel Processing*, vol. 1685 de *Lecture Notes in Computer Science*, p. 677–685. LNCS 1685 - Springer Verlag, 1999.

- [LAT 98] G. LATU. « Solution parallèle pour un simulateur d'un système hôte-parasite en environnement marin », 1998. DEA d'informatique, Université Bordeaux I.
- [LOR 95] H. LOREK et M. SONNENSCHNEIN. « Using parallel computers to simulate individual-oriented models in ecology: A case study ». In *ESM'95 European Simulation Multiconference: Modelling and Simulation*. SCS International, june 1995. p. 526–531.
- [MPI 94] MESSAGE PASSING INTERFACE FORUM. « MPI: A Message-Passing Interface Standard ». *International Journal of Supercomputer Applications*, 8(3-4), 1994. Disponible par ftp anonyme depuis <ftp.netlib.org>.
- [MAN 93] W. MANIATTY, B.K. SZYMANSKI, et T. CARACO. « Implementation and performance of parallel ecological simulations ». In *International Conference on Applications in Parallel and Distributed Computing, Amsterdam, The Netherlands*. Elsevier Science Publishers B.V., 1993.
- [MUR 89] J.D. MURRAY. *Mathematical Biology*. Springer, 2nd édition, 1993.
- [SIL 97] P. SILAN, M. LANGLAIS, et C. BOULOUX. Dynamique des Populations et Modélisation : Application aux Systèmes Hôtes-Macroparasites et à l'épidémiologie en Environnement Marin. In C.N.R.S., Ed., *Tendances nouvelles en modélisation pour l'environnement*. Elsevier, 1997.

Guillaume Latu a obtenu en 1998 un diplôme ingénieur ENSERB ainsi qu'un DEA d'informatique de l'Université Bordeaux I. Il est actuellement allocataire de recherche et prépare une thèse au sein du thème ALiENor du LaBRI (Bordeaux). Ses recherches concernent les techniques algorithmiques dédiées au calcul haute-performance pour la simulation en vraie grandeur d'un système hôte-parasite. Ses travaux sont menés dans un contexte interdisciplinaire impliquant l'épidémiologie, la modélisation mathématique et l'informatique.